cepton_sdk Documentation

Cepton Technologies

Nov 01, 2019

CONTENTS

CHAPTER

ONE

OVERVIEW

If a method is undocumented, consult the C/C++ SDK documentation, since many methods in this library are just wrapper functions.

1.1 Timestamps

Unless otherwise marked, all timestamps are seconds since the Unix epoch (UTC). Note that this differs from the C/C++ interface which uses microseconds.

CHAPTER

TWO

ERRORS

CHAPTER

THREE

SETUP

3.1 Types

3.2 Methods

CHAPTER

FOUR

GENERAL

API for code that is agnostic to live/replay mode.

CHAPTER

FIVE

SENSORS

5.1 Types

5.2 Methods

CHAPTER

SIX

POINTS

6.1 Types

All point array classes support numpy indexing and assignment as if they were 1-d arrays:

```
n_points = len(points_1)
points_2[10:20] = points_1[:10]
```

Multiple point arrays can also be combined:

```
points = cepton_sdk.combine_points([points_1, points_2])
```

6.2 Methods

See *Listen*.

The following methods return points directly from the C SDK callback.

There are also listener classes that seamlessly handle accumulation and waiting.

6.3 Export

CHAPTER

SEVEN

SERIAL

7.1 Methods

The following methods return serial lines directly from the C SDK callback.

There is also a listener class that seamlessly handle accumulation.

CHAPTER

EIGHT

CAPTURE REPLAY

To open/close capture files, use `cepton_sdk.initialize` and `cepton_sdk.deinitialize` methods respectively. The high level API methods will automatically resume the capture replay as necessary.

CHAPTER

NINE

EXPORT

Methods to import/export points to common file formats.

CHAPTER

TEN

SAMPLES

10.1 Multiple Sensors

Listing 1: samples/multiple_sensors.py

```
#!/usr/bin/env python3
"""
Sample script for getting points from multiple sensors simultaneously.
"""

import pprint

import cepton_sdk
import cepton_sdk.plot
from common import *

if __name__ == "__main__":
    # Variables
    capture_path = get_sample_capture_path()

    # Initialize
    cepton_sdk.initialize(capture_path=capture_path, enable_wait=True)

    # Get sensors
    sensors_dict = cepton_sdk.get_sensors()

    # Get points
    listener = cepton_sdk.FramesListener()
    points_dict = listener.get_points()
    del listener
    points_list = next(iter(points_dict.values()))
    points = points_list[0]

    # Plot
    cepton_sdk.plot.plot_points(points)
```

10.2 Single Sensor

Listing 2: samples/single_sensor.py

```
#!/usr/bin/env python3
"""
Sample script for getting points from a single sensor.
"""

import pprint

import numpy

import cepton_sdk
import cepton_sdk.plot
from common import *

if __name__ == "__main__":
    # Variables
    capture_path = get_sample_capture_path()

    # Initialize
    cepton_sdk.initialize(capture_path=capture_path, enable_wait=True)

    # Get sensor
    sensor = cepton_sdk.Sensor.create_by_index(0)
    pprint.pprint(sensor.information.to_dict())

    # Get points
    listener = cepton_sdk.SensorFramesListener(sensor.serial_number)
    points_list = listener.get_points()
    del listener
    points = points_list[0]

    # Plot
    cepton_sdk.plot.plot_points(points)
```

10.3 Advanced

10.3.1 Listen

Listing 3: samples/advanced/listen.py

```
#!/usr/bin/env python3
"""
Sample script for the different methods of getting points.
"""

import numpy

import cepton_sdk
from common import *


def on_frame(serial_number, points):
    print("Received {} points from sensor {}".format(
```

(continues on next page)

(continued from previous page)

```
        len(points), serial_number))


if __name__ == "__main__":
    # Initialize
    cepton_sdk.initialize(capture_path=get_sample_capture_path())
    sensors_dict = cepton_sdk.get_sensors()
    sensor = next(iter(sensors_dict.values()))

    callback_id = cepton_sdk.listen_frames(on_frame)
    cepton_sdk.wait(0.1)
    cepton_sdk.unlisten_frames(callback_id)

    # Get next frames for all sensors. Wait until data is available.
    listener = cepton_sdk.FramesListener()
    points_dict = listener.get_points()
    del listener

    # Get next frames for single sensor. Wait until data is available.
    listener = cepton_sdk.SensorFramesListener(sensor.serial_number)
    points_list = listener.get_points()
    del listener

    # Get large chunk of data
    listener = cepton_sdk.FramesListener()
    cepton_sdk.wait(10)
    points_dict = listener.get_points()
    del listener
    points = cepton_sdk.combine_points(points_dict[sensor.serial_number])
    print("Received {} seconds of data from sensor {}".format(
        numpy.ptp(points.timestamps), sensor.serial_number))
```